(12) UK Patent Application (19) GB (11) 2 362 485 (13) A

(43) Date of A Publication 21.11.2001

(21) Application No 0025253.6

(22) Date of Filing 13.10.2000

(30) Priority Data
(31) 09426248 (32) 25.10.1999 (33) US

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Jennifer Bigus
Daniel Joseph Daley
Richard Alan Diedrich

(74) Agent and/or Address for Service
J P Richards
IBM United Kingdom Limited, Intellectual Property Department, Hursley Park, WINCHESTER, Hampshire, SO21 2JN, United Kingdom

(51) INT CL[7]
G06F 9/44

(52) UK CL (Edition S)
G4A APL

(56) Documents Cited
WO 98/25203 A1 US 5946485 A US 5675753 A
http://www.yrrid.com/LOF/hllapi.htm, Niall Emmart, June 1996
http://www.computerwire.com/cstb/free/214a_18a.htm, Client/Server Technology Jacada 5.0, 12/01/1997

(58) Field of Search
UK CL (Edition S) G4A AKS APL
INT CL[7] G06F 9/44
Online: WPI, EPODOC, PAJ, COMPUTER, IEEE, INSPEC and selected internet sites

(54) Abstract Title
Computer system that defines navigation of a software application

(57) A navigation tool is used to define the flow and operation of a software application 125. The navigation tool receives flow definition information from a user and generates navigation application program interfaces (APIs) that allow a graphical user interface (GUI) 134 to communicate with the software application. These navigation APIs may include flow 132 and operation 133 APIs. These flow and operation APIs preferably interact with preexisting screen scraping APIs 126 to communicate with the software application.

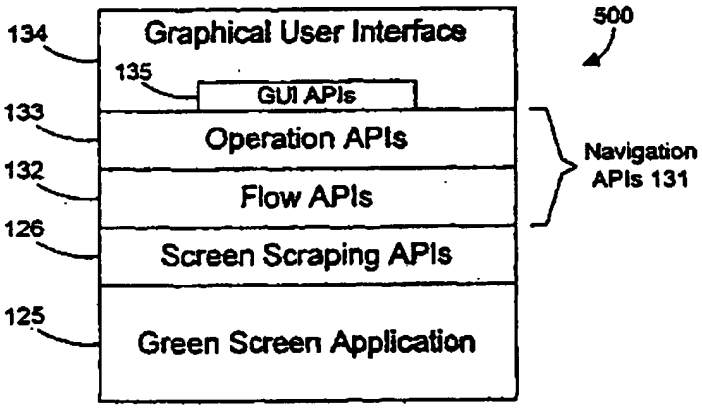

FIG. 5

GB 2 362 485 A

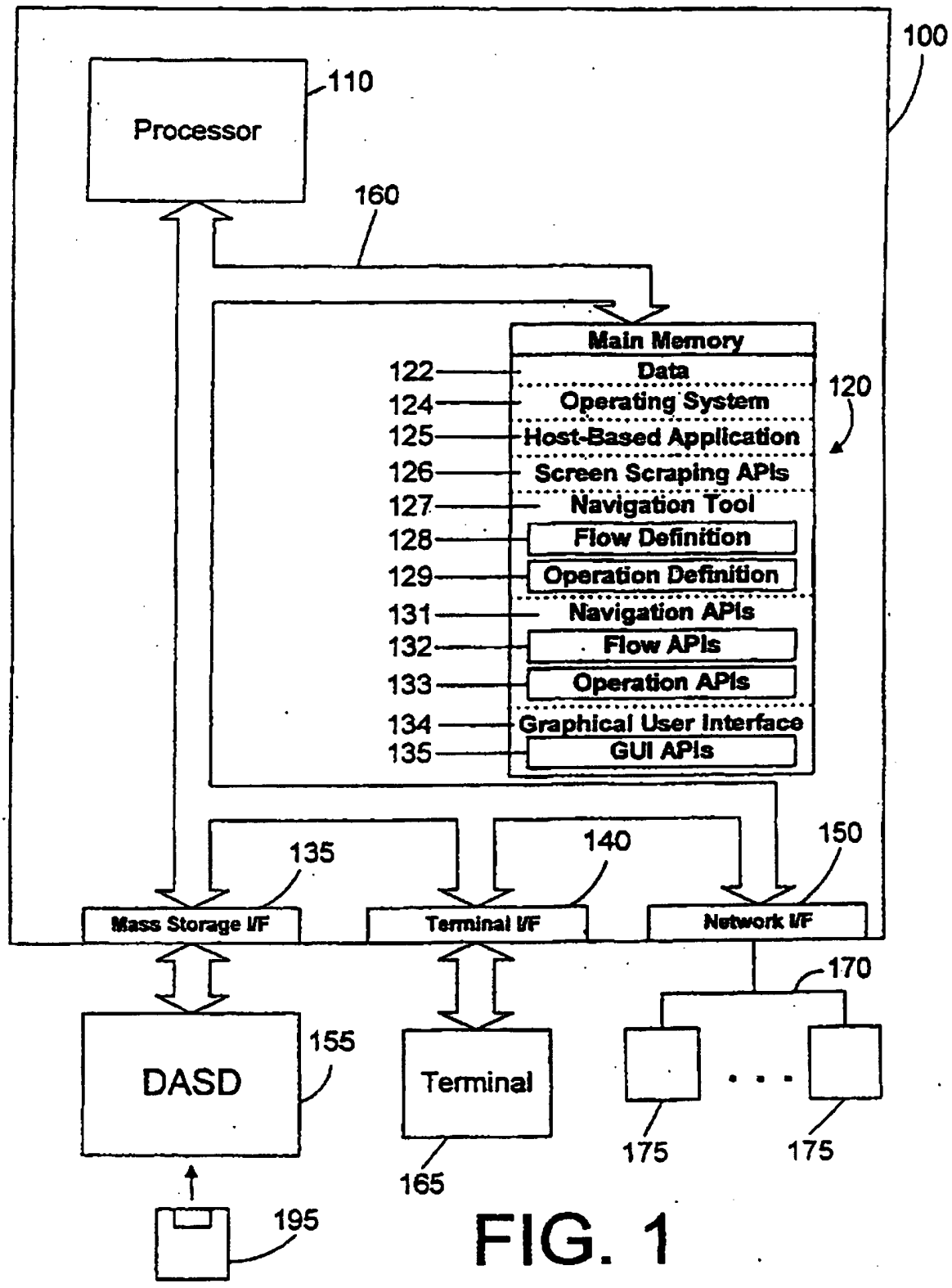

FIG. 1

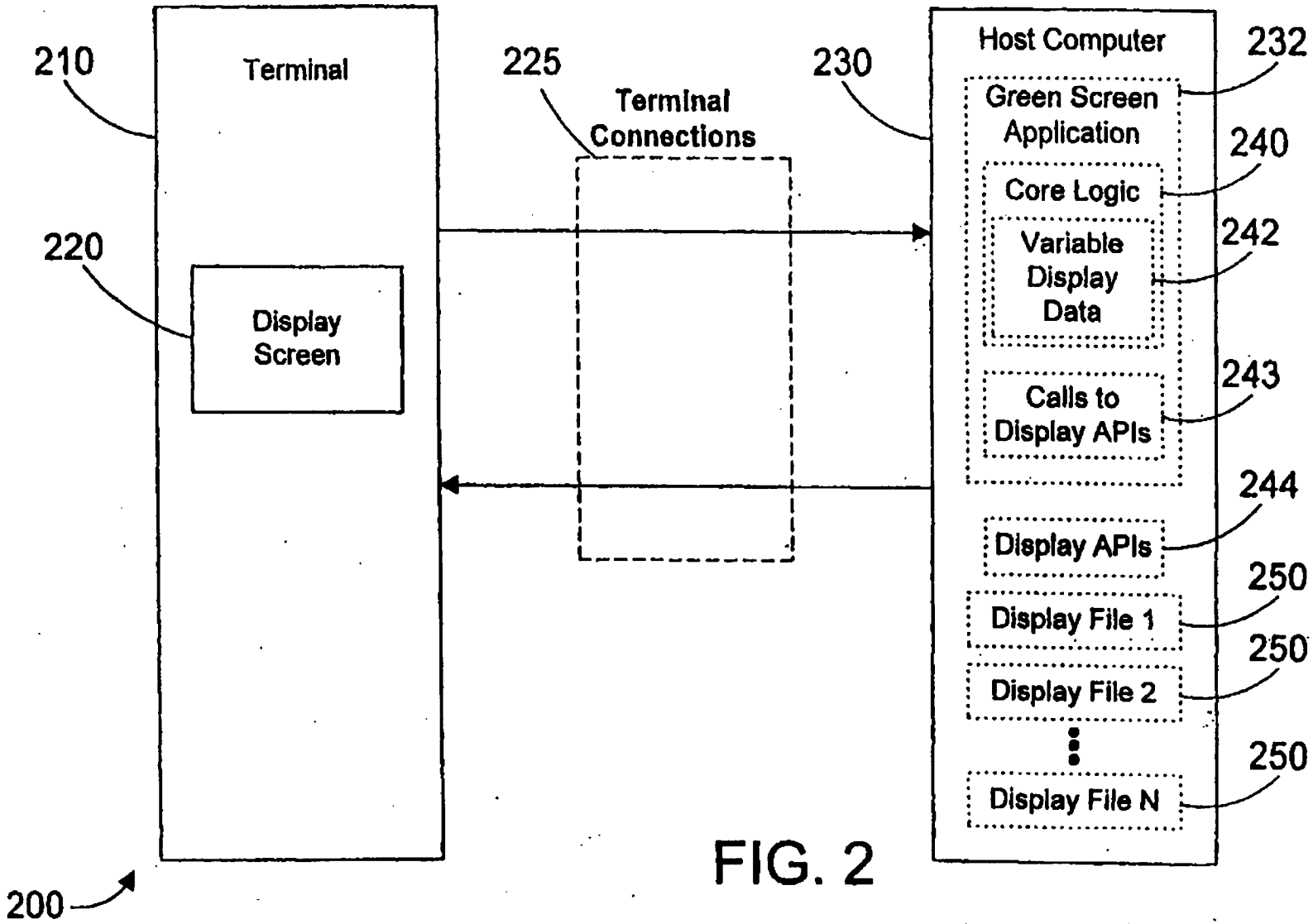
210
Terminal
220
Display Screen
200
225
Terminal Connections
230
Host Computer
232
Green Screen Application
240
Core Logic
242
Variable Display Data
243
Calls to Display APIs
244
Display APIs
250
Display File 1
250
Display File 2
250
Display File N

FIG. 2

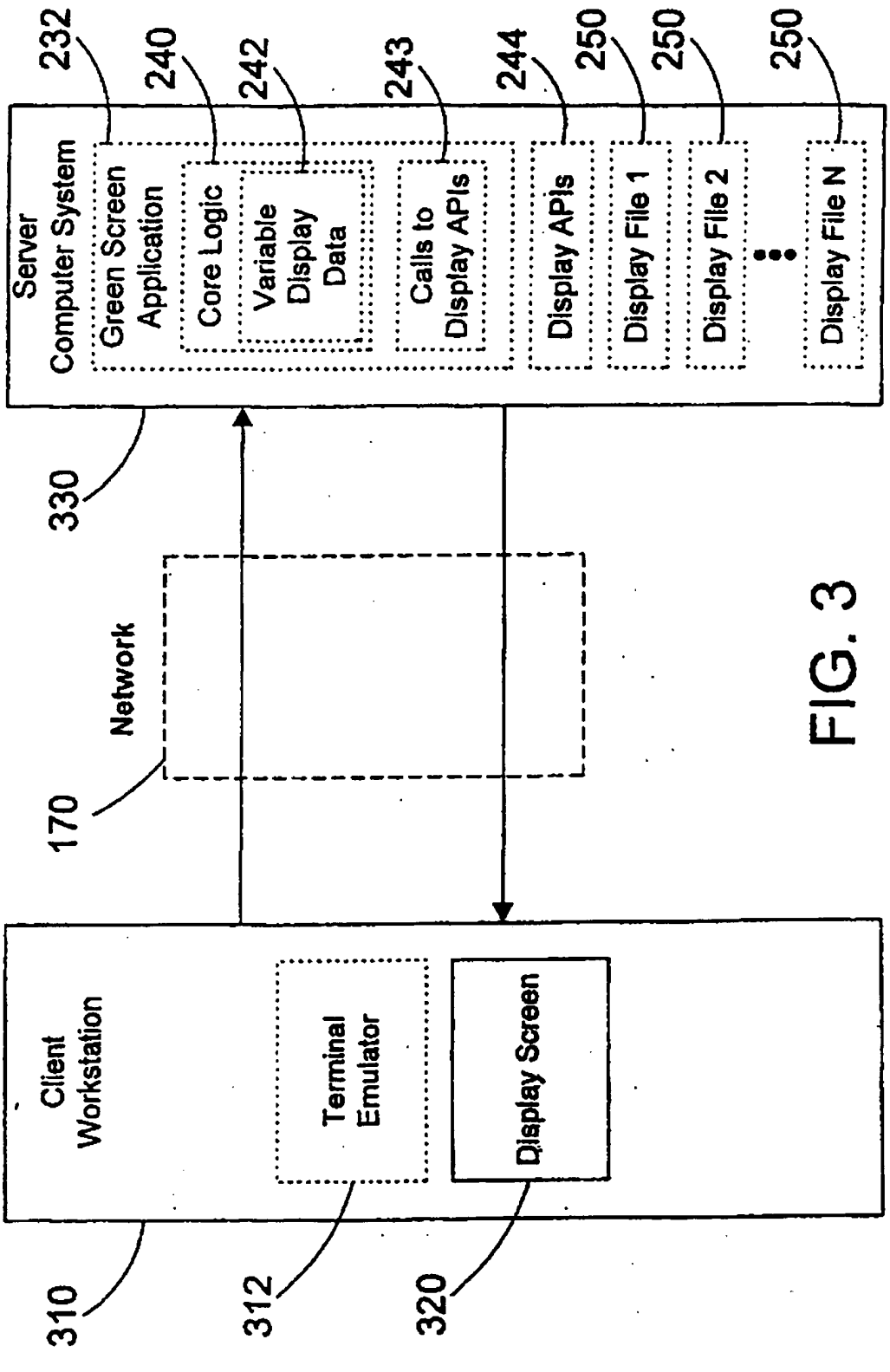

FIG. 3

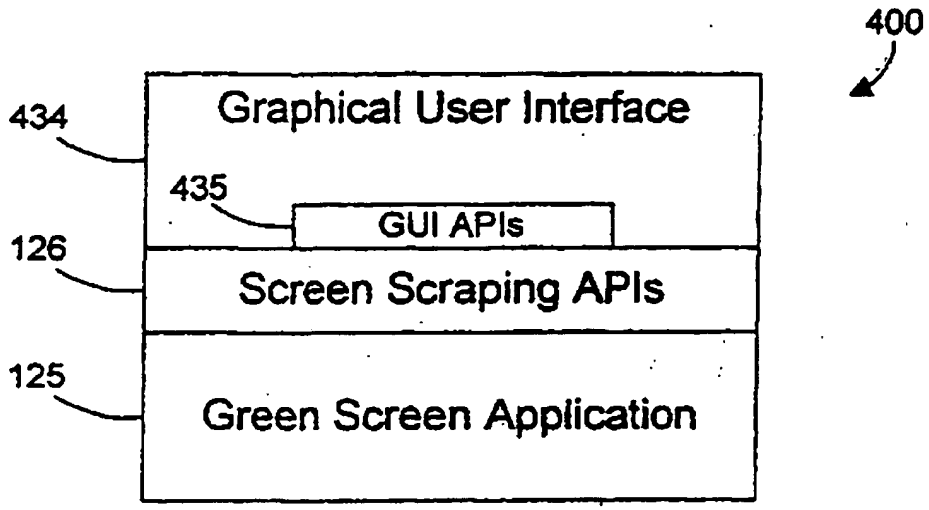

FIG. 4 Prior Art

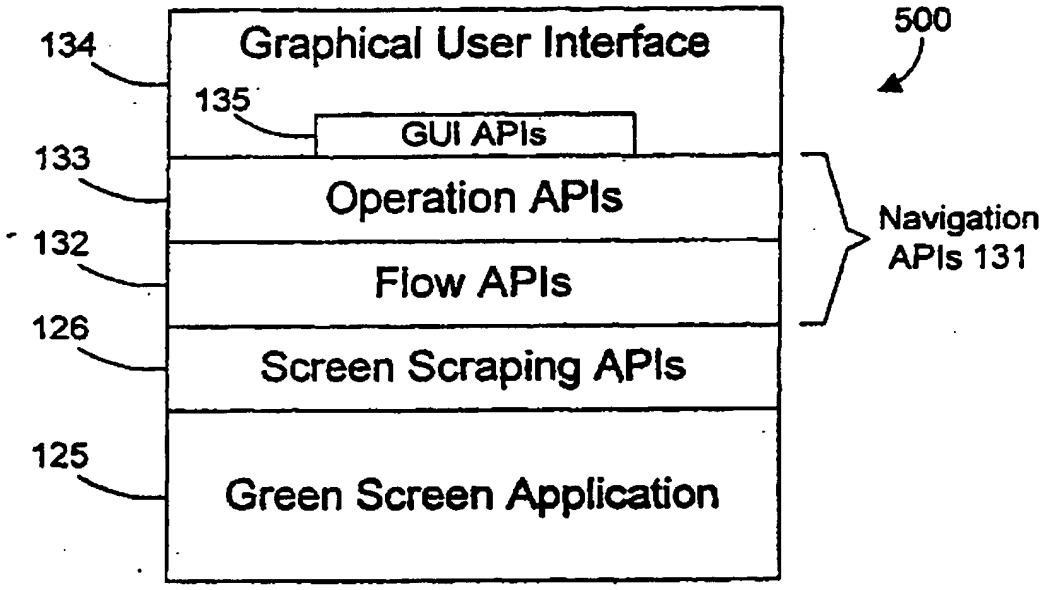

FIG. 5

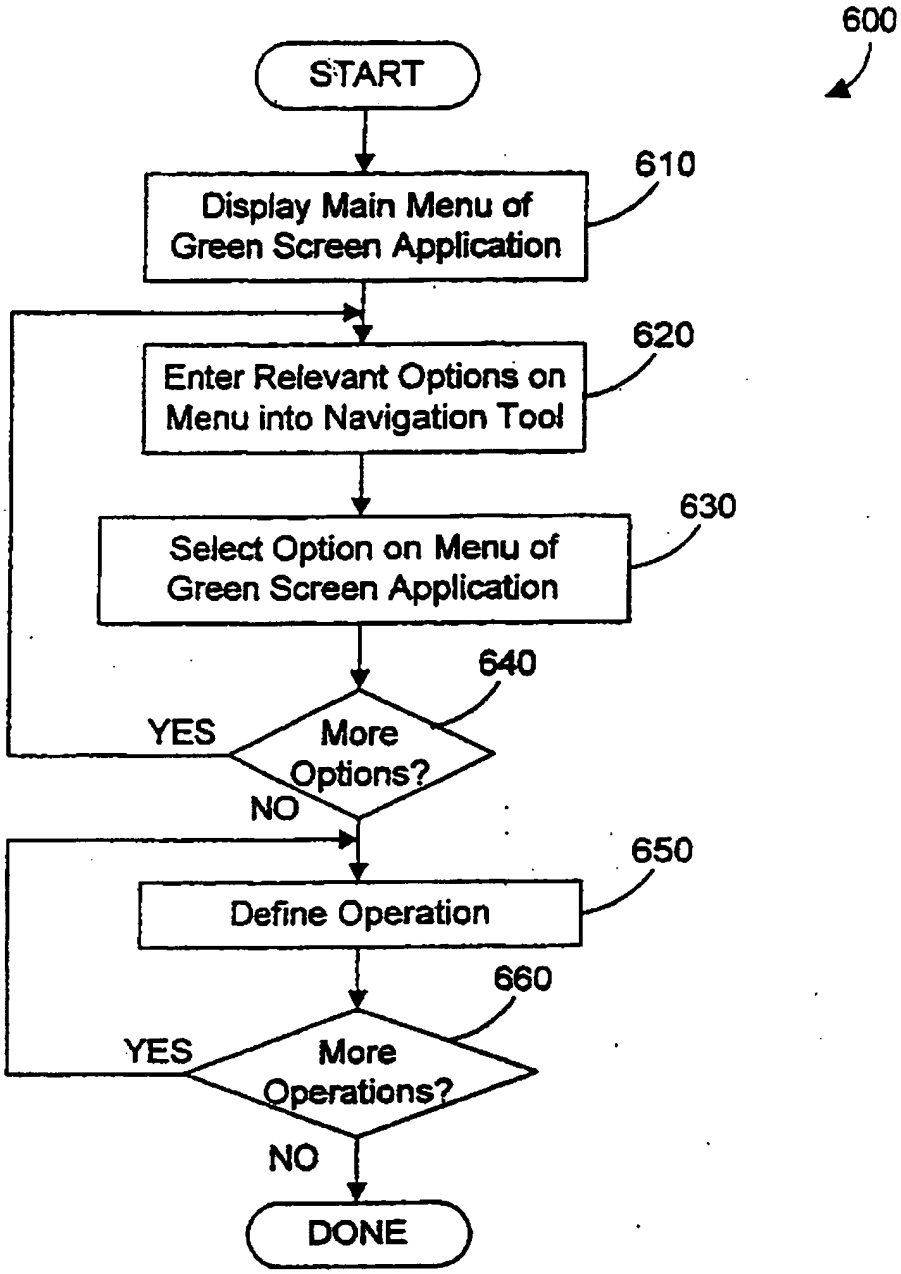

FIG. 6

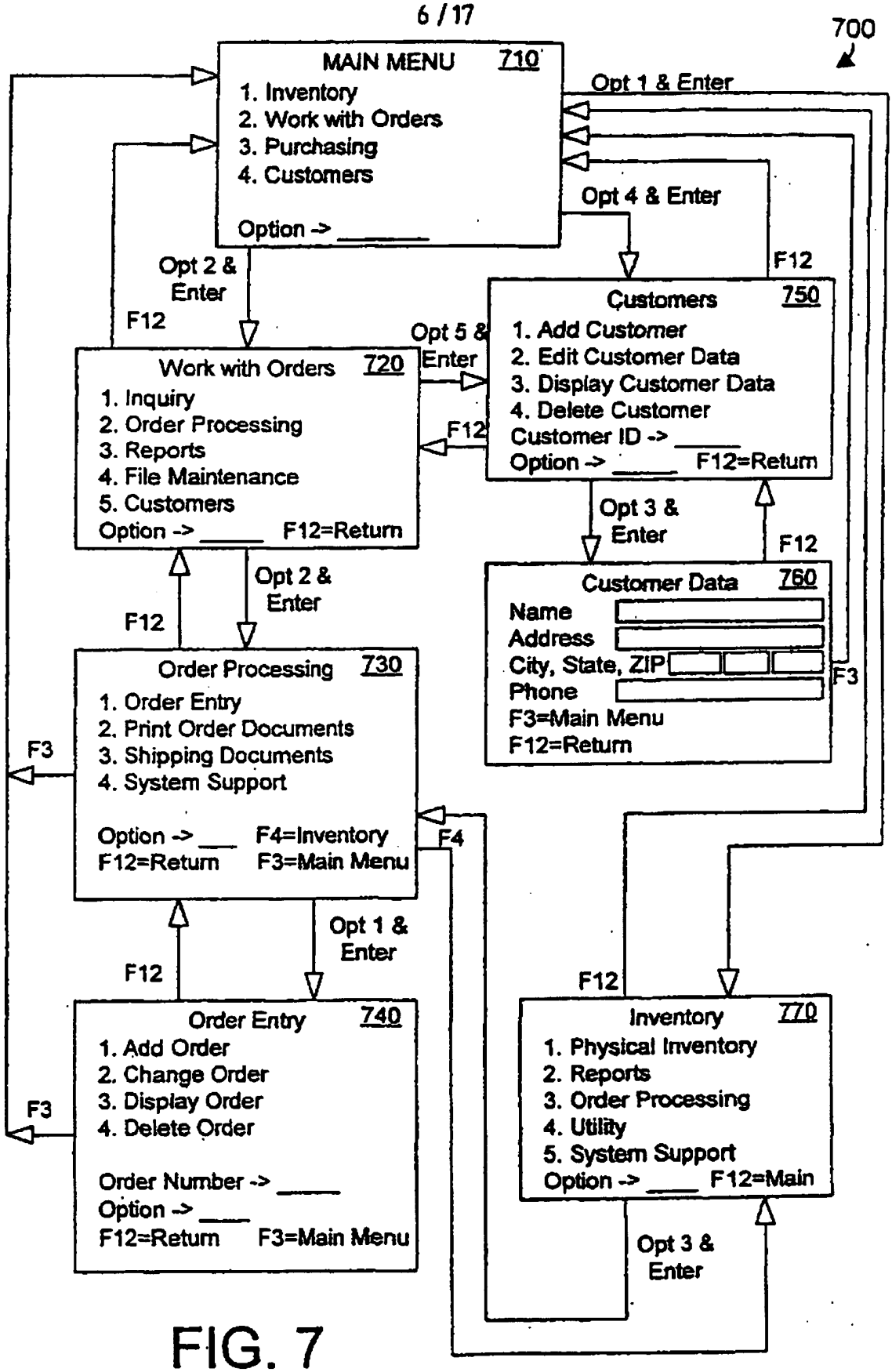

FIG. 7

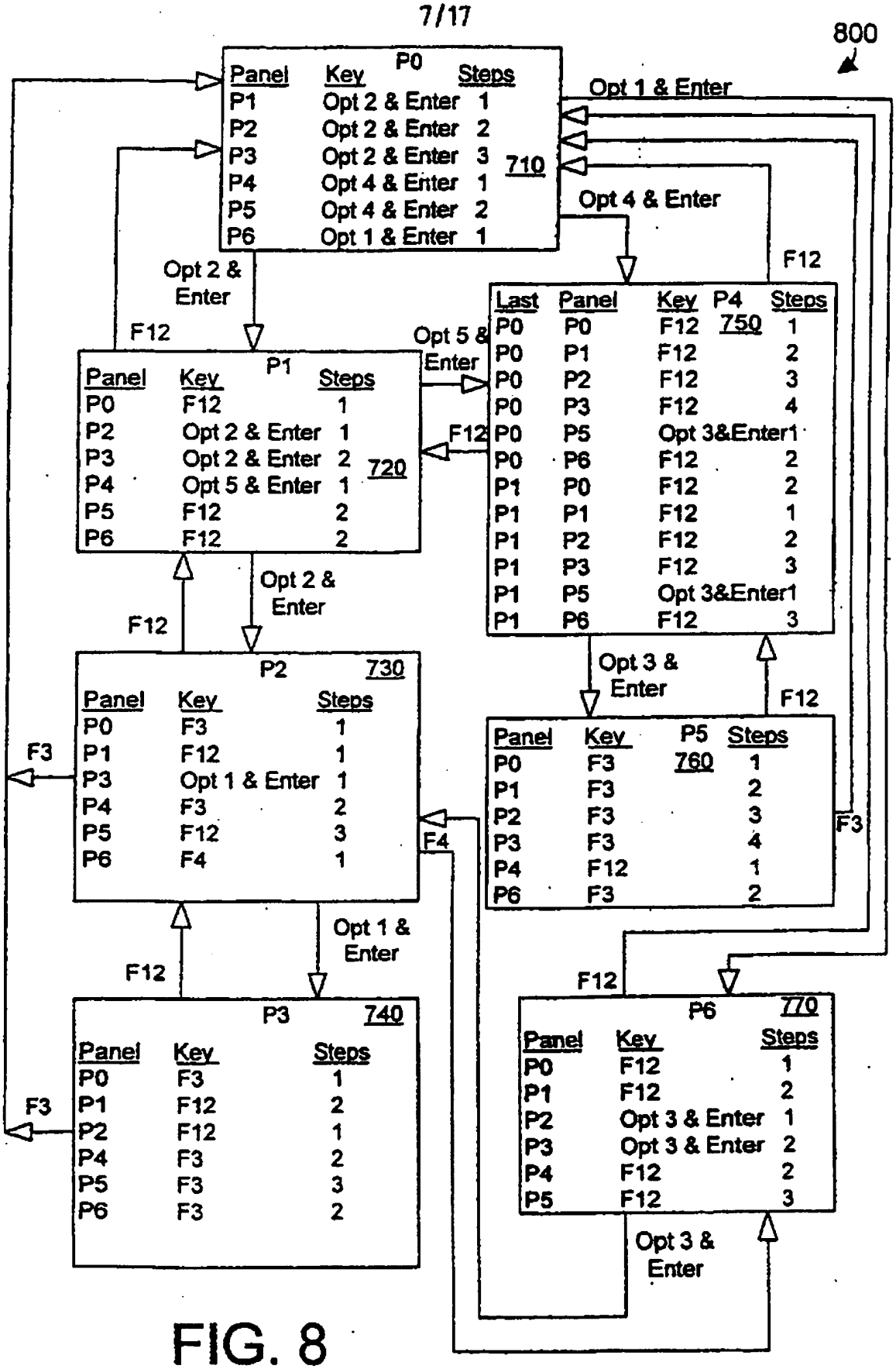

FIG. 8

```
panel:name="MAIN MENU"
        destpanel:name="Inventory",distance="1"
                input:fieldname="Option", value="1"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Work with Orders", distance="1"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Purchasing", distance="1"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customers", distance="1"
                input:fieldname="Option",value="4"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Order Processing", distance="2"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Order Entry", distance="3"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customer Data", distance="2"
                input:fieldname="Option",value="4"
                executekey:name="Enter"
        enddestpanel
endpanel
```

910

920

930

940

950

960

970

FIG. 9

```
panel:name="Work with Orders"
        destpanel:name="Inquiry",distance="1"
                input:fieldname="Option", value="1"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Order Processing", distance="1"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Reports", distance="1"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="File Maintenance", distance="1"
                input:fieldname="Option",value="4"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customers", distance="1"
                input:fieldname="Option",value="5"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Entry", distance="2"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customer Data", distance="2"
                input:fieldname="Option",value="5"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Inventory", distance="2"
                executekey:name="F12"
        enddestpanel
endpanel
```

FIG. 10

```
panel:name="Order Processing"
        destpanel:name="Order Entry",distance="1"
                input:fieldname="Option", value="1"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Print Order Documents", distance="1"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Shipping Documents", distance="1"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="System Support", distance="1"
                input:fieldname="Option",value="4"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Work with Orders", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Customers", distance="2"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Customer Data", distance="3"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Inventory", distance="1"
                executekey:name="F4"
        enddestpanel
endpanel
```

FIG. 11

```
panel:name="Order Entry"
        destpanel:name="Add Order",distance="1"
                input:fieldname="Option", value="1"
                putItem:clientfieldname="OrderNumber",
                        applicationfieldname="OrderNbr"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Change Order", distance="1"
                input:fieldname="Option",value="2"
                putItem:clientfieldname="OrderNumber",
                        applicationfieldname="OrderNbr"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Display Order", distance="1"
                input:fieldname="Option",value="3"
                putItem:clientfieldname="OrderNumber",
                        applicationfieldname="OrderNbr"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Delete Order", distance="1"
                input:fieldname="Option",value="4"
                putItem:clientfieldname="OrderNumber",
                        applicationfieldname="OrderNbr"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Work with Orders", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Processing", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Customers", distance="2"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Customer Data", distance="3"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Inventory", distance="2"
                executekey:name="F12"
        enddestpanel
endpanel
```

FIG. 12

```
panel:name="Customers"
        destpanel:name="Add Customer Data", distance="1"
                input:fieldname="Option",value="1"
                putItem:clientfieldname="custID",applicationfieldname="customerID"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Edit Customer Data", distance="1"
                input:fieldname="Option",value="2"
                putItem:clientfieldname="custID",applicationfieldname="customerID"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customer Data", distance="1"
                input:fieldname="Option",value="3"
                putItem:clientfieldname="custID",applicationfieldname="customerID"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Delete Customer Data", distance="1"
                input:fieldname="Option",value="4"
                putItem:clientfieldname="custID",applicationfieldname="customerID"
                executekey:name="Enter"
        enddestpanel
    if lastpanel:name="MAIN MENU" then:
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Work with Orders", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Processing", distance="3"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Entry", distance="4"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Inventory", distance="2"
                executekey:name="F12"
        enddestpanel
    else /*lastpanel:name="Work with Orders" */
(SEE FIG. 14)
```

FIG. 13

(CONTINUED FROM FIG. 13)

```
        destpanel:name="MAIN MENU", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Work with Orders", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Processing", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Entry", distance="3"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Customer Data", distance="1"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Inventory", distance="3"
                executekey:name="F12"
        enddestpanel
endpanel
```

FIG. 14

```
panel:name="Customer Data"
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Work with Orders", distance="2"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Order Processing", distance="3"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Order Entry", distance="4"
                executekey:name="F3"
        enddestpanel
        destpanel:name="Customers", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Inventory", distance="2"
                executekey:name="F3"
        enddestpanel
endpanel
```

FIG. 15

```
panel:name="Inventory"
        destpanel:name="Physical Inventory",distance="1"
                input:fieldname="Option", value="1"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Reports", distance="1"
                input:fieldname="Option",value="2"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Order Processing", distance="1"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Utility", distance="1"
                input:fieldname="Option",value="4"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="System Support", distance="1"
                input:fieldname="Option",value="5"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="MAIN MENU", distance="1"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Work with Orders", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Order Entry", distance="2"
                input:fieldname="Option",value="3"
                executekey:name="Enter"
        enddestpanel
        destpanel:name="Customers", distance="2"
                executekey:name="F12"
        enddestpanel
        destpanel:name="Customer Data", distance="3"
                executekey:name="F12"
        enddestpanel
endpanel
```

FIG. 16

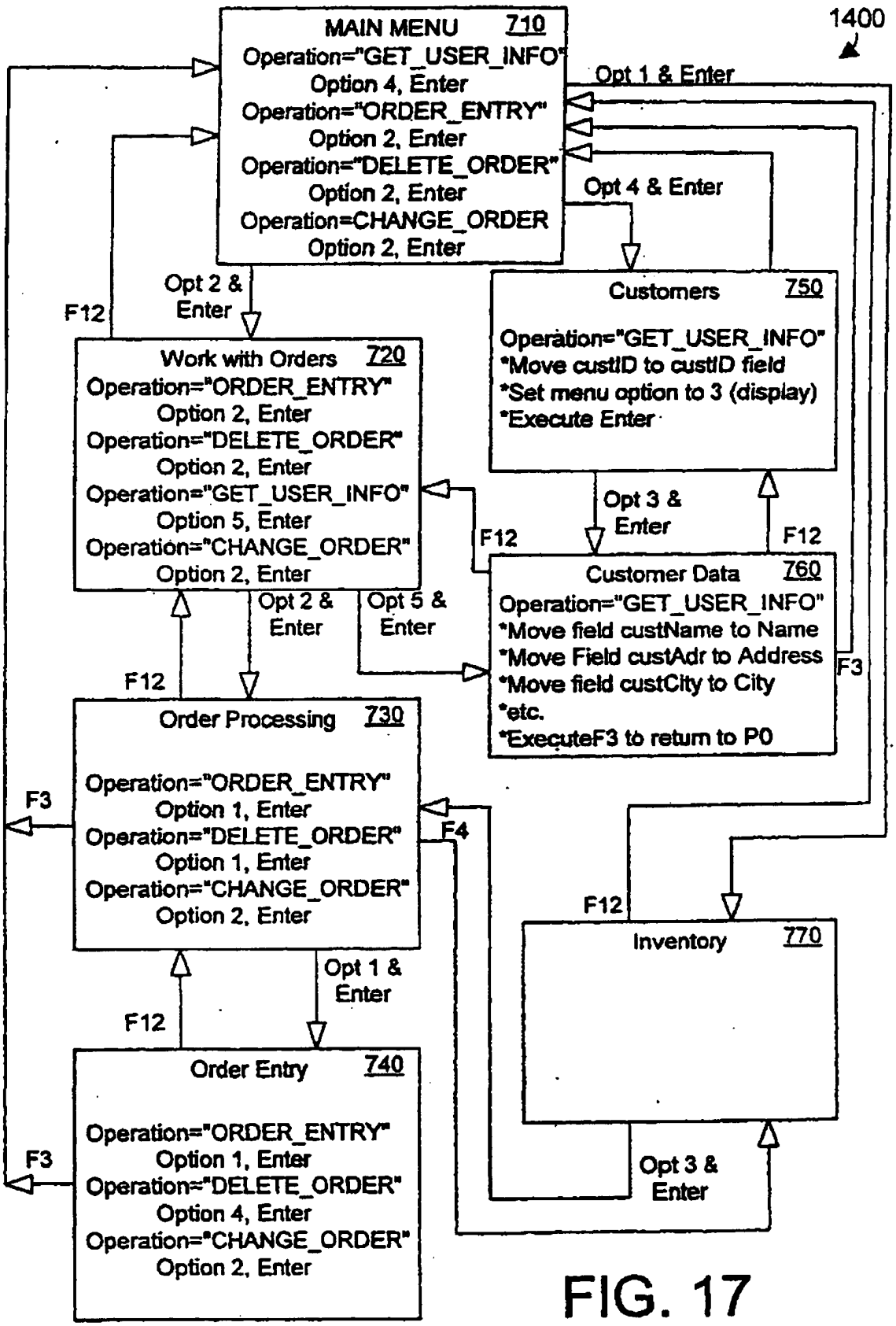

FIG. 17

```
panel:name="MAIN MENU"
        operation:name="GET_USER_INFO"
                input:fieldname="Option", value="4"
                executekey:name="Enter"
        endoperation
endpanel
```

FIG. 18

```
panel:name="Customers"
        operation:name="GET_USER_INFO"
        /*Set the customer number on the application's screen and hit enter*/
          putItem;clientfieldname="custID",applicationfieldname="customerID"
          input:name="Option",value="3"
          executekey:name="Enter"
        endoperation
endpanel
```

FIG. 19

```
panel:name="Customer Data"
      operation:name="GET_USER_INFO"
      /*Read the customer data and send to client*/
            getItem:applicationfieldname="custname",clientfieldname="name"
            getItem:applicationfieldname="custadr1",clientfieldname="address1"
            getItem:applicationfieldname="custadr2",clientfieldname="address2"
            getItem:applicationfieldname="custcity",clientfieldname="city"
            getItem:applicationfieldname="coststate",clientfieldname="state"
            getItem:applicationfieldname="zip",clientfieldname="zip"
            getItem:applicationfieldname="custphone",clientfieldname="phone"
            /*Move back to the main menu*/
            executekey:name="F3"
      endoperation
endpanel
```

FIG. 20

COMPUTER SYSTEM THAT DEFINES NAVIGATION OF A SOFTWARE APPLICATION

This invention generally relates to computer systems, and more specifically relates to data processing in computer systems using a graphical user interface (GUI).

Since the dawn of the computer age, computer systems have evolved into extremely sophisticated devices, and computer systems may be found in many different settings. In the early days of computers, one or more relatively powerful and expensive computers could be shared by many users. A network of computer terminals were typically connected to a single computer known as a "host." These computer terminals are commonly known as non-programmable workstations (i.e., "dumb terminals") because they simply display information transmitted to it by the host, and lack any processing power to perform local tasks. One example of a very well-known computer terminal is the IBM 5250, which displays alphanumeric text in row and column format. In a computing environment with a host and one or more terminals, software applications run on the host, and display information is transmitted by the host to the terminals, which displays an appropriate screen to the user. The user may then enter data in response to the displayed screen, if required. Software applications that ran on these types of systems are known as "host-based" software applications. These are commonly referred to in the computer industry as "green screen" applications, taking their name from the green color of many of the dumb terminals used to interact with these applications.

Green screen applications typically display information in a text-only format of rows and columns on a terminal or computer screen. More modern advances in user interfaces have resulted in a variety of Graphical User Interfaces (GUIs) that allow a user to interact with the computer using a pointing device such as a mouse. A GUI is typically much easier for a user to use compared to a green screen interface. Popular browsers, such as Netscape Navigator (RTM) or Microsoft (RTM) Internet Explorer, include a GUI that accesses and displays information from the world-wide web. A GUI may provide windows, drop-down lists, context-sensitive help, etc., all at the click of a button. Another example of a GUI is the OS/2 (RTM) operating system developed and sold by IBM. As users become more familiar with GUI concepts, entering data into an "old fashioned" green screen application seems less intuitive and training is more tedious. However,

converting all green screen applications to include a GUI would be a formidable and expensive undertaking.

Many green screen applications have been developed for a variety of different industries and companies at great expense. To completely re-write a green screen application to provide a GUI would take considerable time and resources. Not only would it be time-consuming and expensive to re-write a green screen application, but it also would essentially throw away much of the benefit of having stable, solid code that has been debugged and operating correctly for some period of time. A re-write would undoubtedly produce a host of new bugs that would have to be corrected. Thus, what is needed is a way to provide a GUI front-end on a green screen application to avoid re- writing the underlying logic that is tried and true.

One known solution that provides a GUI for green screen applications is known as "screen scraping." With a screen scraping approach, the underlying code of the green screen application is not affected. It still puts and gets data in data streams just as it always did. Screen scraping provides software that takes the screen information and compares the screen information against a table or other database of possible screens for the particular application. Once the screen information is correlated to a known screen, the screen scraping software knows what information needs to be displayed, and can map that information to a GUI. Screen scraping typically employs one or more processes that continually run to provide the translation between green screen format and GUI format. This translation typically requires detecting the screen to be displayed and then displaying the corresponding GUI screen.

With the introduction of the personal computer (PC), networks for personal computers were developed that allow computers to intercommunicate. Thus, instead of providing a host-based system of the past, the personal computer made it possible to have a network of powerful workstations rather than dumb terminals. The processing power of a workstation makes it possible to share processing between computers on a network, rather than having all the processing performed by a host computer. Thus emerged a new paradigm known as client/server computing, or network computing, which allows computer workstations known as clients to interact with one or more server computers.

Migrating green screen applications that were developed for a host-based system into the client/server environment is relatively straightforward. The easiest way is for the computer workstations to

emulate a dumb terminal. While this approach does not take advantage of the processing power of the client workstations, it does allow green screen applications to be implemented in a network computing environment with a minimum of effort. However, as described above, users in a network computing environment are becoming more accustomed to GUIs for their applications, and would prefer to interact with green screen applications using a GUI.

Known screen scraping techniques require a programmer to custom-program a GUI to each green screen application by calling low-level functions that interact with the green screen application. The programmer must have detailed knowledge of the application and the GUI, and create code that bridges the gap between the two by defining the navigation of the green screen application for the GUI. Without a way to define navigation of a software application without custom-programming a GUI, the computer industry will continue to suffer from excessive development costs when creating a GUI for a green screen application.

According to the present invention there is provided a computer system comprising:

at least one processor;

a memory coupled to the at least one processor;

a navigation tool residing in the memory for execution by the at least one processor, the navigation tool receiving flow definition for a software application from a user and generating therefrom at least one application program interface (API) for navigating the software application.

In an embodiment of the invention, a navigation tool is used to define the flow and operation of a host-based application. The navigation tool generates from the flow and operation information navigation APIs that allow a GUI to communicate with the host-based application. In the preferred embodiments, these navigation APIs include flow APIs and operation APIs. These flow and operation APIs preferably interact with pre-existing screen scraping APIs to communicate with the host-based application. By defining higher level functions in the flow APIs and operation APIs than those available in the screen scraping APIs, the GUI may be programmed in a more general and descriptive way that makes porting the GUI to a different host-based application much easier to do. The navigation tool of the present invention is especially useful for web-enabling an existing host-based application. The user enters the relevant information from the host-based application into the navigation tool, which then generates navigation APIs for navigating the host-based application. Any functions of the host-based application that are not entered into the navigation tool will not have corresponding APIs generated, which makes it easy to define which functions will be web-enabled (by providing support in the GUI) and which functions will not.

Embodiments of the present invention will now be described with reference to the accompanying drawings, wherein like designations denote like elements, and wherein:

FIG. 1 is a block diagram of an apparatus in accordance with a preferred embodiment of the present invention;

FIG. 2 is a block diagram of a host computer system running a green screen application;

FIG. 3 is a block diagram of a client/server network running a green screen application;

FIG. 4 is a block diagram of a prior art computer program;

FIG. 5 is a block diagram of a computer program that includes navigation APIs in accordance with the preferred embodiments of the present invention;

FIG. 6 is a flow diagram of a method for defining navigation of an application in accordance with a preferred embodiment of the present invention;

FIG. 7 is block diagram of one example menu structure for a sample green screen application for illustrating the concepts of the preferred embodiment;

FIG. 8 is a block diagram of the display panels shown in FIG. 7 with notations relating to the navigation of the green screen application;

FIG. 9 is sample script description of the "Main Menu" panel shown in FIG. 7;

FIG. 10 is sample script description of the "Work with Orders" panel shown in FIG. 7;

FIG. 11 is sample script description of the "Order Processing" panel shown in FIG. 7;

FIG. 12 is sample script description of the "Order Entry" panel shown in FIG. 7;

FIGS. 13 and 14 are a sample script description of the "Customers" panel shown in FIG. 7;

FIG. 15 is sample script description of the "Customer Data" panel shown in FIG. 7;

FIG. 16 is sample script description of the "Inventory" panel shown in FIG. 7;

FIG. 17 is a block diagram of the menu structure of FIG. 7 showing the definition of operations in accordance with the preferred embodiments;

FIG. 18 is a sample script description of a GET_USER_INFO operation defined in the Main Menu panel of FIG. 7;

FIG. 19 is a sample script description of a GET_USER_INFO operation defined in the Customers panel of FIG. 7; and

FIG. 20 is a sample script description of a GET_USER_INFO operation defined in the Customer Data panel of FIG. 7.

Referring to FIG. 2, a host-based computer system 200 includes a dumb terminal 210 that is coupled via terminal connections 225 to a host computer 230. Dumb terminal 210 includes a display screen 220 for displaying information to a user. Host computer 230 includes a green screen application 232. Green screen application 232 includes core logic 240 that contains and/or generates variable display data 242. Green screen application 232 communicates with terminal 210 through calls 243 to display application program interfaces (APIs) 244. Display APIs 244 typically include a PUT API, a GET API, and a PUTGET API. The PUT API outputs display information to terminal 210 for display on display screen 220. The GET API receives input information from terminal 210. The PUTGET API outputs display information to terminal 210 for display on display screen 220, then waits for the user to respond with input information.

Data is output for display on terminal 210 by passing variable display data 242 to one of the PUT or PUTGET APIs through a call 243 to the appropriate API. The API then determines which display file 250 is specified. The display file may be specified along with the variable display data, but more commonly is specified by a separate FORMAT command that identifies the appropriate display file 250 before passing the variable display data 242 to the PUT or PUTGET API. The information to be output to terminal 210 typically includes a variable portion, represented by variable display data 242, combined with a fixed portion, represented by a corresponding display file 250. The combination of variable and fixed data is a screen or a portion of a screen to be displayed on terminal 210.

The advent of the PC and computer networks for PCs created client/server environments. Referring to FIG. 3, a client/server computer system 300 may be used in the place of the host/terminal computer system 200 of FIG. 2. Green screen applications are easily ported to a client/server environment 300 by placing the green screen application 232 in a server computer system 330. The green screen application itself 232, display APIs 244 and display files 250 remain unchanged, but are now executed by a server that serves as a host. Network 170 replaces terminal connections 225, and a client workstation 310 replaces dumb terminal 210. Client workstation 310 is preferably a computer system (such as a PC) that can perform local processing as well as communicating with other computers via network 170. Client workstation 310 typically includes a terminal emulator 312 that makes client workstation 310 appear as a dumb terminal to any programs (such as green screen application 240) that format their display input and output for dumb terminals. In this manner, a

client/server environment 300 may be made to emulate the host/terminal environment 200 of FIG. 2.

As stated in the Background section, screen scraping is a known technique that allows a graphical user interface (GUI) to be defined for a green screen application. One of the principal advantages of screen scraping is that the logic of the green screen application is unchanged. As far as the green screen application is concerned, it is still communicating with a dumb terminal. Screen scraping software thus provides low-level application program interfaces (APIs) that emulate the dumb terminal interface for the green screen application in function calls for a graphical user interface. Examples of known screen scraping software include IBM's Enhanced High Level Language Application Program Interface (EHLLAPI), and object oriented APIs provided by Jacada, Inc.

Screen scraping APIs can best be understood with reference to a known computer program 400 shown in FIG. 4. A green screen application 125 represents any software application that is designed to communicate with a dumb terminal. Screen scraping APIs 126 (such as those referenced above provided by IBM Corp. or Jacada, Inc.) provide low- level functions that put to and get data from the green screen application 125. Graphical user interface 434 is a GUI that is custom-programmed to invoke screen scraping APIs 126 when data exchange with the green screen application 125 is required. GUI 434 typically includes a number of predefined GUI APIs 435 that can be used along with screen scraping APIs 126 to communicate between the green screen application 125 and GUI 434. Screen scraping APIs 126 and GUI APIs 435 thus provide a translation mechanism for moving data between the green screen application 125 and the GUI 434. Note, however, that the graphical user interface 434 must include logic that allows directly invoking the screen scraping APIs 126, which requires that GUI 434 have detailed knowledge of how the screen scraping APIs 126 interact with green screen application 125. The result is a GUI that is custom programmed for a single green screen application, which cannot be easily ported to a different green screen application.

According to a preferred embodiment of the present invention, a software tool is used to define the navigation of a green screen application, and the tool then uses this information to generate navigation APIs (*e.g.*, flow APIs and operation APIs) that allow a GUI to invoke these APIs to communicate with the green screen application without the GUI having any detailed knowledge regarding the green screen application.

Referring to FIG. 5, the APIs that are generated by the software tool in accordance with the preferred embodiments are referred to herein as flow APIs 132 and operation APIs 133, which are collectively referred to as navigation APIs 131. Note that these APIs constitute "middleware", which is used herein to refer to software that is used to communicate between two other software programs or program portions. In FIG. 5, the screen scraping APIs 126 and green screen application 125 are the same as shown in the prior art in FIG. 4. The primary distinction over the prior art is the addition of the flow APIs 132 and operation APIs 133, which result in a greatly simplified graphical user interface 134. Flow APIs 132 define functions at a higher level than screen scraping APIs 126, such as how the navigation of the green screen application is performed. Flow APIs 132 suitably invoke screen scraping APIs 126 to define these navigation functions. Operation APIs 133 define functions at a higher level than flow APIs 132. Operation APIs 133 suitably invoke flow APIs 132 to define operations that may need to be performed by interacting with the green screen application 125. Graphical user interface 134 is thus programmed by calling high level functions defined by flow APIs 132 and operation APIs 133, instead of dealing with the low-level screen scraping APIs 126. These navigation APIs 131 thus provide high-level flow and operation functions that may be invoked by the graphical user interface, which effectively removes the need for any information specific to the screen scraping APIs 126 in developing the graphical user interface 134. Note that in the preferred embodiments, GUI 134 includes a number of GUI APIs 135 that can be used to pass data to and from GUI 134.

Referring to FIG. 1, computer system 100 is an enhanced IBM AS/400 computer system. However, those skilled in the art will appreciate that the mechanisms and apparatus of the present invention apply equally to any computer system, regardless of whether the computer system is a complicated multi-user computing apparatus, a single user workstation, or an embedded control system. As shown in FIG. 1, computer system 100 comprises a processor 110 connected to a main memory 120, a mass storage interface 135, a terminal interface 140, and a network interface 150. These system components are interconnected through the use of a system bus 160. Mass storage interface 135 is used to connect mass storage devices (such as a direct access storage device 155) to computer system 100. One specific type of direct access storage device is a floppy disk drive, which may store data to and read data from a floppy diskette 195.

Main memory 120 contains data 122, an operating system 124, a host-based application 125, screen scraping APIs 126, a navigation tool

127, navigation APIs 131, and a graphical user interface 134. Navigation tool 127 includes a portion for flow definition 128 and a portion for operation definition 129. In addition, navigation APIs 131 include flow APIs 132 and operation APIs 133, and GUI 134 includes GUI APIs 135. Computer system 100 utilizes well known virtual addressing mechanisms that allow the programs of computer system 100 to behave as if they only have access to a large, single storage entity instead of access to multiple, smaller storage entities such as main memory 120 and DASD device 155. Therefore, while data 122, operating system 124, host-based application 125, screen scraping APIs 126, navigation tool 127, navigation APIs 131, and GUI 134 are shown to reside in main memory 120, those skilled in the art will recognize that these programs are not necessarily all completely contained in main memory 120 at the same time. It should also be noted that the term "memory" is used herein to generically refer to the entire virtual memory of computer system 100. Note that in the preferred embodiments of the invention, different parts of main memory 120 are distributed among multiple computer systems on a network. For example, in a client/server system similar to system 300 of FIG. 3, computer system 100 would have a server computer system running host-based application 125, while a client computer system would have the screen scraping APIs 126, navigation tool 127, navigation APIs 131, and GUI 134.

Processor 110 may be constructed from one or more microprocessors and/or integrated circuits. Processor 110 executes program instructions stored in main memory 120. Main memory 120 stores programs and data that processor 110 may access. When computer system 100 starts up, processor 110 initially executes the program instructions that make up operating system 124. Operating system 124 is a sophisticated program that manages the resources of computer system 100. Some of these resources are processor 110, main memory 120, mass storage interface 135, terminal interface 140, network interface 150, and system bus 160.

Data 122 represents any data that serves as input to or output from any program in computer system 100. Operating system 124 is a multitasking operating system known in the industry as OS/400; however, those skilled in the art will appreciate that the present invention is not limited to any one operating system.

Host-based application 125 is any software application that formats its data input and output for a dumb terminal. A green screen application is one suitable example of a host-based application. Screen scraping APIs

126 are low-level application program interfaces (*i.e.*, programs) that are called to get data from and send data to the host-based application 125.

Graphical user interface 134 is a GUI that is designed to communicate with the host-based application 125 by invoking navigation APIs 131. GUI 134 suitably includes a number of GUI APIs that provide an architected interface for communicating with GUI 134. This allows the details of the screen scraping APIs 126 to be encapsulated within the higher level flow APIs 132 and operation APIs 133, thus giving the programmer much higher level navigation and operation capabilities without knowing how the screen scraping APIs perform their functions. This results in a GUI that can be ported to different green screen applications with a minimum of effort by defining navigation APIs for the different green screen application that implement the functions called by an existing GUI.

Navigation tool 127 includes a portion for flow definition 128 and a portion for operation definition 129. The navigation tool 127 is a software tool that is used by a programmer to enter information relating to the navigation of a host-based application. Typically, the programmer invokes the main menu on the host-based application, and enters information that is displayed on the main menu into the navigation tool 127 to define how the application may be navigated. Navigation tool 127 may present a graphical user interface that allows the user to define the flow definition 128 and operation definition 129. In the alternative, tool 127 may read one or more script files that describe the flow definition 128 and operation definition 129. Of course, there are many other ways for a user to define flow definition 128 and operation definition 129. The flow definition 128 and operation definition 129 of navigation tool 127 are used to generate flow APIs 132 and operation APIs 133, respectively. These navigation APIs 131 define high level functions that may be invoked by the GUI 134 without interacting directly with the screen scraping APIs 126.

Although computer system 100 is shown to contain only a single processor and a single system bus, those skilled in the art will appreciate that the present invention may be practiced using a computer system that has multiple processors and/or multiple buses. In addition, the interfaces (called input/output processors in AS/400 terminology) that are used in the preferred embodiment each include separate, fully programmed microprocessors that are used to off-load compute-intensive processing from processor 110. However, those skilled in the art will appreciate that the

present invention applies equally to computer systems that simply use I/O adapters to perform similar functions.

Terminal interface 140 is used to directly connect one or more terminals 165 to computer system 100. These terminals 165, which may be non-intelligent (*i.e.*, dumb) terminals or fully programmable workstations, are used to allow system administrators and users to communicate with computer system 100. Note, however, that while terminal interface 140 is provided to support communication with one or more terminals 165, computer system 100 does not necessarily require a terminal 165, because all needed interaction with users and other processes may occur via network interface 150.

Network interface 150 is used to connect other computer systems and/or workstations (*e.g.*, 175 in FIG. 1) to computer system 100 across a network 170. The present invention applies equally no matter how computer system 100 may be connected to other computer systems and/or workstations, regardless of whether the network connection 170 is made using present-day analog and/or digital techniques or via some networking mechanism of the future. In addition, many different network protocols can be used to implement a network. These protocols are specialized computer programs that allow computers to communicate across network 170. TCP/IP (Transmission Control Protocol/Internet Protocol) is an example of a suitable network protocol.

At this point, it is important to note that while the present invention has been (and will continue to be) described in the context of a fully functional computer system, those skilled in the art will appreciate that the present invention is capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution. Examples of suitable signal bearing media include: recordable type media such as floppy disks (*e.g.*, 195 of FIG. 1) and CD ROM, and transmission type media such as digital and analog communications links.

The remainder of this specification describes how navigation tool 127 is used to generate flow definition 128 and operation definition 129 of host-based application 125, to generate corresponding flow APIs 132 and operation APIs 133 that can be invoked by GUI 134. Navigation tool 127 thus provides a way to easily generate middleware (*e.g.*, flow APIs 132 and operation APIs 133), which provide much higher level functions to GUI 134 than those provided by screen scraping APIs 126. As a result, GUI 134 can

be more easily ported to a different green screen application by using navigation tool 127 to define APIs for the different green screen application that are called by GUI 134.

Referring now to FIG. 6, method 600 shows steps in accordance with the preferred embodiments for entering flow definition 128 and operation definition 129 into navigation tool 127. First, the main menu (*i.e.*, starting menu) of the green screen application is displayed (step 610) on one computer system. A user then reads the menu options that the main menu presents, and enters information for relevant options into the navigation tool (step 620) that is running on a different computer system or in a different window on the same computer system. Note that "relevant options" as used herein means options that the user wants to enable in the graphical user interface. This concept is especially useful in the context of web-enabling an existing green screen application. The relevant options that are entered in step 620 are those options that the programmer wants to web-enable, meaning those that need to be accessible via the world-wide web. Many companies have sophisticated green screen applications that perform mission-critical functions that should not be accessible to outsiders, as well as performing other functions (such as order processing) that needs to be accessible via the world-wide web. Navigation tool 127 allows the user to select which features are relevant, and should therefore be included in the GUI. All options that are not entered into the navigation tool 27 will not be accessible via the GUI. Step 620 thus provides a way to easily define which options should be enabled in the GUI by defining those options in the navigation tool, while all other options will not be accessible via the GUI.

Once all selected options on the main menu are entered into the navigation tool, one of the relevant options is selected (step 630) on the computer system that is running the green screen application. The result is that the next menu is presented to the user. If this menu contains relevant options that need to be enabled in the graphical user interface (step 640=YES), method 600 returns to step 620, and steps 620, 630 and 640 are repeated until all of the relevant options that are to be enabled in the graphical user interface are entered into the navigation tool (step 640=NO). At this point the navigation tool has all the relevant information relating to the flow between menus in the green screen application, and the user can now define higher level operations (step 650) that use the flow information to generate functions that greatly simplify the design of the graphical user interface. Once all desired operations are defined (step 660=NO), method 600 is done. At this point the menu

options entered by the user make up the flow definition 128 of FIG. 1, and the definitions of the operations entered by the user make up the operation definition 129. Once the navigation tool 127 performs the steps in method 600 of FIG. 6, it then suitably generates the flow APIs 132 and operation APIs 133 that may be invoked by graphical user interface 134 (FIG. 1).

A sample menu structure 700 for an imaginary green screen application is shown in FIG. 7. Each box corresponds to a screen (*e.g.*, menu display, display panel, or data entry panel) in the green screen application, and the arrows between boxes illustrate how a user may move between menus. The main menu 710 includes four options: 1) inventory; 2) work with orders; 3) purchasing; and 4) customers. The user selects one of these options by pressing the number corresponding to the option and pressing the Enter key. When the second option is selected, the work with orders menu 720 is displayed to the user. On this menu, the user can select one of five options, or may return to the previous menu by pressing the F12 key. Selecting the second option "order processing" in the work with orders menu 720 results in displaying the order processing menu 730, which has four different options that may be selected. In addition to the listed options, pressing the F4 key displays the inventory menu 770, pressing the F12 key displays the work with orders menu, and pressing the F3 key displays the main menu, as shown by the key definitions at the bottom of the order processing menu 730. If the order entry option in the order processing menu 730 is selected, the order entry menu 740 is then displayed. Order entry menu 740 includes four options. In addition, the order processing menu can be displayed by pressing the F12 key, and the main menu can be displayed by pressing the F3 key. Selecting any of the options on the order entry menu 740 would display other menus that are not shown for the sake of simplicity. The sample menu structure 700 is shown to illustrate some of the features of the preferred embodiments of the present invention.

When the main menu 710 is displayed, selecting the fourth option results in displaying the customers menu 750, which provides four options. Pressing the F12 key returns to the main menu 710 or the work with orders menu 720, depending on which one was the last menu displayed before invoking the customers menu 750. Because the customers menu 750 may be displayed from both the main menu 710 and the work with orders menu 720, pressing the F12 key to "return" must result in returning to the menu that was displayed just prior to displaying the customers menu 750, which will be one of the menus 710 or 720. When any of options 1, 2, or 4 in the customers menu 750 is selected, another screen (not shown) is displayed to perform those functions. For the purpose of illustration, only one of the

four options are represented in FIG. 7. When the user selects option 3 and presses the Enter key, the customer data entry screen 760 is displayed, which includes data fields for the customer's name, address, city, state, ZIP, and phone number. In addition, the main menu may be displayed by pressing F3, and pressing F12 results in returning to the customers menu 750.

When the main menu is displayed, selecting the first option results in displaying the inventory menu 770, which includes five options. In addition, when displaying the inventory menu 770, the main menu may be displayed by pressing the F12 key.

The information in the menu structure 700 of FIG. 7 can be used to derive all information needed to navigate between menus in the green screen application. This navigation information is shown by the menu structure of FIG. 8. Each menu is assigned a panel number from P0 to P6. Each menu contains three columns that list the display panels (*i.e.*, menus) that can be displayed, the key or keys that are pressed to select that panel, and how many steps away the selected panel is from the current menu. For example, the main menu is assigned a label of P0 (panel 0). The information in the columns shows that to reach display panel P1 (which corresponds to the work with orders menu 720), option 2 is selected followed by pressing the Enter key, and this panel is only one step away. Navigating to display panel P2 (which corresponds to the order processing menu 730) from the main menu is done by first navigating to display panel P1, and then navigating to display panel P2, because there is no direct route from the main menu to the order processing menu 730. The entry for P2 in P0 thus shows the same key sequence as for P1, but that P2 is two steps away. In similar fashion, panel P3 (which corresponds to the order entry menu 740) can only be visited from the main menu by passing through P1 and P2. For this reason, the entry in P0 for P3 shows the same key sequence (*i.e.*, Opt 2 & Enter) as the P1 and P2 entries, but P3 is three steps away.

The entry for display panel P4 in P0 indicates that P4 is visited from P0 by pressing option 4 and Enter on the main menu, and is one step away. Panel P5 can only be visited by navigating through P4, so the entry for display panel P5 in P0 shows the same key strokes as P4, namely Opt 4 & Enter, but is two steps away. Finally, the entry for display panel P6 in P0 shows that navigating to P6 from P0 is done by pressing Opt 1 & Enter, and is one step away. For each menu, the keys that navigate to other menus that are one step away are shown by arrows leading from the menu. To

navigate to menus that are more than one step away, one of the arrows out of the menu is taken, which will navigate one step closer to the desired menu. Note that the preferred embodiments prefer the shortest path to longer paths, so if multiple paths to a menu exist, only the shortest path is represented. In some circumstances, there may be multiple shortest paths. When this occurs, the selection of one of the shortest paths may be random, or may be done according to any suitable selection heuristic.

Referring now to panel P1 (720) of FIG. 8, when panel P1 (which corresponds to the work with orders menu 720 shown in FIG. 7) is selected, the user may navigate to the main menu (panel P0) that is only one step away by pressing the F12 key. Panel P2 can be displayed by pressing Opt 2 & Enter, and is only one step away. Navigating to panel P3 requires passing through P2, so the entry for P3 in P1 shows the same keys to get to panel P2 (namely, Opt 2 & Enter), but P3 is two steps away instead of one. Navigating to panel P4 is done by pressing Opt 5 and Enter, and is one step away. Navigating to panel P5 and P6 is done by pressing the F12 key, and both of these panels are two steps away (through P0).

Panel P2 of FIG. 8 is considered next. Panel P2 corresponds to the order processing menu 730 of FIG. 7. To navigate from P2 to P0, the F3 key is pressed, and P0 is one step away. To navigate from P2 to P1, which is one step away, the F12 key is pressed. Navigating to panel P3 is done by pressing Opt 1 & Enter, and P3 is one step away. Navigating to panel P4 is done by pressing F3 (to go through P0), and is two steps away. Navigating to panel P5 is done by pressing F12 (to return to P1), and is three steps away. Navigating to panel P6 is done by pressing F4, and is one step away.

Referring now to panel P3 in FIG. 8 (which corresponds to the Order Entry menu 740 of FIG. 7), to navigate from P3 to P0, the F3 key is pressed, and P0 is only one step away. To navigate to P1, the F12 key is pressed (to go through P2), and P1 is two steps away. To navigate to P2, the F12 key is pressed, and P2 is one step away. To navigate to P4, the F3 key is pressed, and P4 is two steps away (through P0). To navigate to P5, the F3 key is pressed, and P5 is three steps away (through P0 and P4). To navigate to P6, the F3 key is pressed, and P6 is two steps away (through P0).

Panel P4 of FIG. 8 corresponds to the customers menu 750 of FIG. 7. Note that panel P4 in FIG. 8 contains a column "Last" that does not exist in any of the other panels. This column is required, because the navigation information varies when the F12 key is pressed to return to the previous menu depending on the last menu before navigating to P4. In other

words, if the user navigates to P1, then to P4 and presses the F12 key, P4 will return to P1 (the last panel in time that navigated to P4). If, on the other hand, the user navigates to P0 then to P4, pressing F12 will return to P0 from whence it came. Because the F12 key can result in navigating to different panels depending on which panel was the last in time to navigate to panel P4, the "last" panel information is needed to navigation from panel P4. If the last panel was P0, navigation to P0 may be done by pressing the F12 key, and panel P0 is only one step away. If the last panel was P0, navigation to P1 is done by pressing the F12 key, which returns to P0, and P1 is two steps away from P4. In similar fashion, if the last panel was P0, navigation to panels P2 and P3 is also done by pressing the F12 key, with P2 being three steps away and P3 being four steps away. Navigation to P5 does not depend on the last panel, so the entries for P5 shows that regardless of whether P0 or P1 was the last panel, navigating from P4 to P5 is done by pressing the Opt 3 & Enter keys, and P5 is one step away. If the last panel was P0, navigating to P6 is done by pressing the F12 key, and P6 is two steps away. If the last panel was P1, the steps in navigating to P0, P1, P2, P3, and P6 are different than the case above, when the last panel was P0. Navigating to each of these screens from P4 when the last panel was P1 is done by pressing the F12 key, but the number of steps are different than the number of steps when P0 was the last panel, because pressing F12 results in navigating to P1 instead of P0.

Panel P5 of FIG. 8 corresponds to the customer data entry screen 760 of FIG. 7. Navigating to panel P0 is done by pressing the F3 key, and P0 is one step away. Navigating to panel P1 is done by pressing the F3 key, and P1 is two steps away (through P0). Navigating to panel P2 and P3 are done in similar fashion, by pressing the F3 key, with the panel P2 being three steps away and the panel P3 being four steps away. Navigating to panel P4 is done by pressing the F12 key, and P4 is only one step away. Navigating to panel P6 is done by pressing the F3 key, and P6 is two steps away.

Referring again to FIG. 8, panel P6 corresponds to the inventory menu 770 of FIG. 7. Navigating to panel P0 is done by pressing the F12 key, which returns to P0 which is one step away. Navigating to panels P1 and P4 is done by pressing the F12 key, and P1 and P4 are both two steps away (through P0). Navigating to panel P2 is done by pressing the Opt 3 & Enter keys, and P2 is only one step away. Navigating to panel P3 is done by pressing the Opt 3 & Enter keys, and P3 is two steps away (through P2).

Navigating to P5 is done by pressing the F12 key, and P5 is three steps away.

The menu structure in FIG. 7 is a sample menu structure that is suitably generated by the navigation tool 127 of FIG. 1 in accordance with the preferred embodiments. The information used to this menu structure can be entered by a user in a number of different ways within the scope of the preferred embodiments. In one embodiment of method 600 of FIG. 6, a user navigates the green screen application on one system and enters relevant options and information into the navigation tool 127 using a graphical user interface. Once all relevant options have been entered into the navigation tool 127, the graphical user interface for the navigation tool suitably produces a graphical menu structure for the green screen application, similar to that shown in FIG. 7. In another embodiment, a user can define the navigation of a green screen application in a suitable script language. Examples of sample script language definitions for the menus in FIG. 7 are shown in FIGS. 9-16. For all the script language definitions in FIGS. 9-16, the "panel" keyword defines the start of a panel definition. The "destpanel" keyword defines which menus can be reached from this menu and the distance (*i.e.*, number of steps) to the destination screen. The "input" keyword is used to set the value of a field on the screen. The "executekey" keyword is used to specify the key to be pressed to cause the application to take action and move to the next menu. Typically, these are function keys or the enter key. The "distance" is the number of steps to the selected menu.

Referring to FIG. 9, a script language definition for the main menu 710 is shown. Portion 910 shows that when the destination panel is the inventory panel, navigating to the inventory panel is done by entering option 1 and pressing enter. Portion 920 shows that when the destination panel is the work with orders panel, navigating to the work with orders panel is done by entering option 2 and pressing enter. In similar fashion, portions 930-970 each define how to navigate from the main menu to the other destination panels.

FIGS. 10-16 each show a script language definition for menus defined in FIG. 7. FIG. 10 defines the navigation from the work with orders panel 720 to each of the other panels. In addition, FIG. 10 also shows the navigation to each of the relevant options on the work with orders menu, even though the menus that correspond to these options are not shown in FIG. 7. For example, in the script language definition of the work with orders menu 720 in FIG. 10, panels "Inquiry", "Reports", and "File

Maintenance" are defined in the script language even though these menus are not explicitly shown in FIG. 7 for the sake of simplicity.

FIG. 11 shows a script language description of the order processing menu 730. Each of the relevant options in the order processing menu 730 and each of the other menus are included as destination panels, with the appropriate key strokes and distance to navigate to or towards the destination panel.

FIG. 12 shows a script language description of the order entry menu 740 of FIG. 7. FIGS. 13 and 14 show a script language description of the customers menu 750 of FIG. 7. FIG. 15 shows a script language description of the customer data entry screen 760 of FIG. 7. FIG. 16 shows a script language description of the inventory menu 770 of FIG. 7. The script language descriptions in FIGS. 9-16 could be used by navigation tool 127 to define the navigation of the green screen application, as shown in graphical form in FIG. 8.

In the preferred embodiments, once the navigation of the green screen application is defined, navigation tool 127 generates one or more flow APIs (132 in FIG. 1) that may be invoked to navigate the user interface for the green screen application. Once the flow APIs have been generated, higher level operations can also be defined that use the flow APIs 132 to accomplish a desired task. Referring now to FIG. 17, a menu structure 1400 similar to the menu structure of FIG. 8 is displayed, but instead of showing the navigation information, operations are defined for each menu. For example, in main menu 710 of FIG. 17, the following operations are defined: GET_USER_INFO, ORDER_ENTRY, DELETE_ORDER, and CHANGE_ORDER. Each operation definition includes a specification of the key or keys that cause the operation to be executed. The work with orders menu 720 defines these same four operations, but the keys for executing these operations are different than those defined in the main menu 710. The order processing menu 730 includes an ORDER_ENTRY operation, a DELETE_ORDER operation, and a CHANGE_ORDER operation, with their corresponding keys to invoke these operations. The order entry menu 740 defines these same operations, but the keys used to invoke the DELETE_ORDER operation are different than those used to invoked the DELETE_ORDER operation in the order processing menu 730. The customers menu 750 of FIG. 17 defines a GET_USER_INFO operation, and how this operation is executed. In similar fashion, the customer data entry screen 760 defines a GET_USER_INFO operation, and how this operation is executed. Note that the inventory menu 770 does not have any defined

operations. We assume for this example that inventory operations are not enabled in the graphical user interface.

In the preferred embodiments, once operations are defined (as shown in FIG. 17), navigation tool 127 generates operation APIs (133 of FIG. 1) for each defined operation. These operation APIs 133 invoke flow APIs 132 as needed to navigate the green screen application in performing the desired operation. One significant advantage to defining operations in this manner is that a graphical user interface with different characteristics of the green screen application can be defined. For example, a green screen application may be designed to communicate with a user by opening a session as each new user logs on, performing the desired functions, and closing the session when the user is done. Opening and closing sessions in a green screen application are functions that require significant system overhead. It may be preferable to open a session, and to allow for information from multiple users to be entered into the green screen application during a single session. The capability of defining operations for each menu thus allows the capabilities of the green screen application to be expanded beyond what its native interface would allow.

Similar to navigation, operations for a menu can be defined using a graphical user interface to navigation tool 127, can be defined in a suitable script language, or can be defined in any other suitable manner. FIGS. 18-20 show suitable script language for the GET_USER_INFO operation shown in FIG. 17. Referring to FIG. 18, the operation GET_USER_INFO" for the main menu is defined. This operation is executed by pressing the Opt 4 & Enter keys. Referring to FIG. 19, a GET_USER_INFO operation on the Customers menu is performed by putting the value in the customerID field of the green screen application in the custID field of the client field (i.e., GUI). Once the customer ID is entered, option 3 is selected, and the Enter key is pressed. Referring to FIG. 20, a GET_USER_INFO operation on the customer data panel is performed by placing the customer's name stored in the custname field of the green screen application in the "name" field of the GUI; by placing the first part of the customer's address in the custadr1 field of the green screen application in the "address1" field of the GUI; by placing the second part of the customer's address in the custadr2 field of the green screen application in the "address2" field of the GUI; by placing the customer's city in the custcity field of the green screen application in the "city" field of the GUI; by placing the customer's state in the custstate field of the green screen application in the "state" field of the GUI; by placing the customer's ZIP code in the zip field of the green screen application in the "zip" field of the GUI; by

placing the customer's phone number in the custphone field of the green screen application in the "phone" field of the GUI; and by executing the F3 key to move back to the main menu 710. One suitable way for placing data in fields in the GUI is to invoke GUI APIs 135 within the GET_USER_INFO operation. However, there are other ways within the scope of the preferred embodiments for performing operations and exchanging data with the GUI. For example, instead of putting the interaction with GUI APIs 135 within the GET_USER_INFO operation, the GUI itself could perform the GET_USER_INFO operation, which retrieves the data, and the GUI could then perform explicit functions (such as calls to GUI APIs and/or additional APIs) to move the retrieved data into the GUI. This an other variations for communicating between the GUI and the green screen application are expressly within the scope of the preferred embodiments.

The embodiments described herein disclose an apparatus, method, and program product that define one or more navigation APIs that can be called to navigate a software application. In the preferred embodiments, a graphical user interface is defined that calls these APIs, thus allowing the GUI to rely on higher level functions that are independent of any particular green screen application. The present invention is especially useful when web-enabling a green screen application. Operation APIs define the highest level functions that may be called by the GUI. These APIs in turn call flow APIs that define the flow in the green screen application. The operation and flow APIs can both invoke screen scraping APIs to communicate with the green screen application. By providing this "middleware" that provides another level of abstraction between the screen scraping APIs and a GUI, the programmer of the GUI need not have any information regarding the specific screen scraping APIs that are used to access the green screen application. Instead, access to the screen scraping APIs is preferably encapsulated by invoking only operation APIs and flow APIs in the GUI, which in turn may access screen scraping APIs as appropriate. As a result, a GUI that is designed to call the navigation APIs of the present invention can easily be ported to a different green screen application by using navigation tool 127 to generate flow APIs and operation APIs that are called by the existing GUI.

CLAIMS

1. A computer system comprising:

at least one processor;

a memory coupled to the at least one processor;

a navigation tool residing in the memory for execution by the at least one processor, the navigation tool receiving flow definition for a software application from a user and generating therefrom at least one application program interface (API) for navigating the software application.

2. The system of claim 1 wherein the software application comprises a host-based application.

3. The system of claim 1 wherein the navigation tool further receives operation definition for the software application from the user and generates therefrom at least one API.

4. The system of claim 1 wherein the flow definition is defined using a graphical user interface.

5. The system of claim 1 wherein the flow definition is defined in a script language.

6. The system of claim 1 wherein the at least one API comprises at least one flow API that at least partially defines flow of the software application.

7. The system of claim 1 wherein the at least one API comprises at least one operation API that defines at least one operation that is performed by interacting with the software application.

8. The system of claim 1 further comprising a plurality of screen scraping application program interfaces (APIs) residing in the memory, wherein the at least one API for navigating the software application invokes at least one of the screen scraping APIs.

9. A method for providing a plurality of application program interfaces (APIs) that may be called by a graphical user interface (GUI) to navigate a software application, the method comprising the steps of:

(1) invoking a display panel in the software application;

(2) entering information about the display panel into a navigation tool;

(3) repeating steps (1) and (2) until all desired information relating to the software application has been entered into the navigation tool; and

(4) the navigation tool generating from the information entered in step (2) the plurality of APIs.

10. A method for providing a graphical user interface for a software application, the method comprising the steps of:

a user entering flow information for the software application into a navigation tool; and

the navigation tool generating from the flow information at least one application program interface (API) that is called by the graphical user interface to navigate the software application.

11. A program product comprising:

(A) a navigation tool that receives flow definition for a software application from a user and generates therefrom at least one application program interface (API) for navigating the software application; and

(B) signal bearing media bearing the navigation tool.

Application No: GB 0025253.6 **Examiner:** Adam Tucker
Claims searched: 1-11 **Date of search:** 11 September 2001

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.S): G4A APL, AKS

Int Cl (Ed.7): G06F 9/44

Other: Online: WPI, EPODOC, PAJ, COMPUTER, IEEE, INSPEC and selected internet sites

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	WO98/25203 A1	JBA HOLDINGS PLC, See pp 25-27	
A	US 5946485	WEEREN et al., See abstract and abstract Fig. and summary of the invention cols 2-4	-
A	US 5675753	HANSEN et al., See cols 7-9	-
X	http://www.yrrid.com/LOF/hllapi.htm, Niall Emmart, June 1996		1-7, 9-11
A	http://www.computerwire.com/cstb/free/214a_18a.htm, Client/Server Technology Jacada 5.0, 12/01/1997		-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

BNSDOCID: <GB_____2362485A_I_>